COBB

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS Versions 5 & 6

August 1994
Vol. 5 No. 8

VERSION 6.x

Running MSBACKUP from the command line

We should all back up our systems on a regular basis—that's a given. If you've upgraded to DOS 6.x and you're one of the faithful who makes regular backups, by now you might be somewhat frustrated with the MSBACKUP utility. It's great the first few times you make backups because the help system, menus, and dialog boxes help you through the process and remind you of all your options.

However, if you back up with any regularity, you'd probably prefer to avoid all the hand-holding—by now you're pretty familiar with the options on the Backup menu. You know exactly which files you want to back up, where you want to back them up, which options you use, and whether you want a full, incremental, or differential backup. If you're in this situation, we're pleased to report that this article will show you how to back up your files from the command line using MSBACKUP with an undocumented switch.

The technique

To run MSBACKUP from the command line, you have to specify a setup file; so if you don't have a setup file, you'll have to create one in the MSBACKUP utility. You select the files you want to back up regularly, specify the backup type, and select certain options from the Disk Backup Options menu. Then, you save and name the setup file.

Once you've created a setup file, you can return to the DOS prompt and run MSBACKUP with the command

```
msbackup setup /batch
```

where *setup* is the name of your setup file. You don't need to include the file extension. The backup will run automatically and will pause only to prompt you to exchange diskettes. When all the files are backed up, the utility will end and return to the DOS prompt.

We created a setup file for all the files we want to back up monthly. This setup file, which we named MONTHLY.SET, includes our batch, INI, graphics, text, database, and spreadsheet files—in other words, all the files not stored on other disks. Now it's your turn to create and modify a setup file to use at the command line.

Creating a monthly setup file to use at the command line

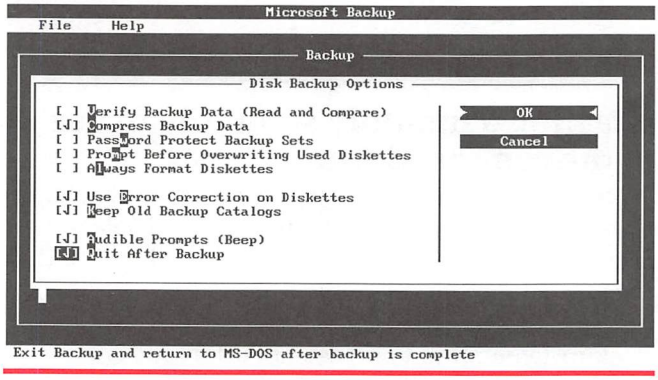
To create your monthly setup file, you select Backup from the Microsoft Backup 6.0 menu and select the files you want to back up each month. Then use the Backup To item to select the floppy drive to which you want to back up your files. Next, choose the Backup Type option and select Full. Now, choose Options... to open the Disk Backup Options dialog box shown in Figure A on the next page. Choose the options you want, but be sure to remove the check mark from the Prompt Before Overwriting Used Diskettes box and place a check mark in front of the Quit After Backup option. (You add or remove a check mark by clicking the check box with your mouse or by using the arrow keys to move the cursor to the check box and pressing the [Spacebar].) Finally, click OK or press [Enter].

Once you return to the Backup dialog box, you save the setup file by pressing [Alt]F and then A for the Save Setup As... command. Type the name *MONTHLY.SET* in the File Name text box, type *Full backup--nonexecutables* in the Description text box, and then highlight Save and press [Enter]. Your Backup dialog box will look similar to the one

IN THIS ISSUE

shown in Figure B. Notice that next to the Select Files... button, MSBACKUP shows how many files you've chosen to back up, tells how many floppy disks you'll need, and estimates how long the backup will take.

Figure A

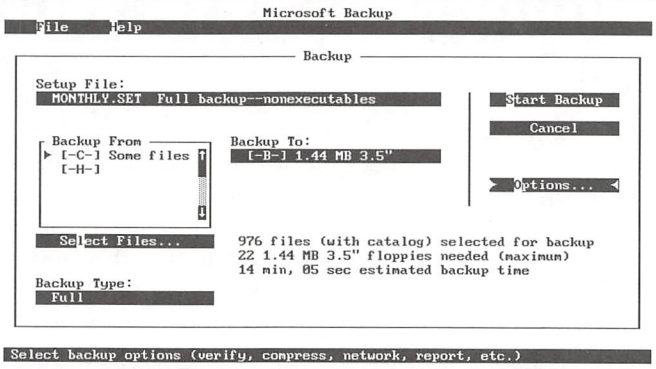


Deselect the Prompt Before Overwriting Used Diskettes option and select the Quit After Backup option.

Modifying MONTHLY.SET to use as a daily setup file

To create a daily setup file, you simply modify the monthly file. You open the Backup dialog box and select the MONTHLY.SET file in the Setup File text box. Next, select the Backup Type. Choose Incremental if you want to keep copies of your files as they change—in other words, if you plan to store your daily backup disks as well as your monthly backup disks. Choose Differential to back up only the files that you've changed since the last full backup—in other words, if you plan to store only the most recent daily backup disks as well as the monthly backup disks.

Figure B



After you save your setup file, the Backup dialog box shows the choices you've made and estimates how many disks you'll need and how long the backup will take.

You already set the Options... you want to use and selected the files you want to back up when you created the monthly setup file. All you have left to do is save the file. Open the Save Setup File dialog box by

INSIDE DOS

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices: Domestic: $49/yr ($6.00 each); Outside US: $69/yr ($8.50 each)

Phone: Toll free: (800) 223-8720; Local: (502) 493-3300; Customer Relations Fax: (502) 491-8050; Editorial Department Fax: (502) 491-3433

Address: You may address tips, special requests, and other correspondence to

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster: Second class postage is pending in Louisville, KY. Send address changes to
Inside DOS
P.O. Box 35160
Louisville, KY 40232

Authorized Canada Post International Publications Mail (Canadian Distribution) Sales Agreement #XXXXXX CANADA GST #123669673. Send returns to Canadian Direct Mailing Sys. Ltd., 920 Mercer Street, Windsor, Ontario, N9A 7C2. Printed in the USA.

Staff:

Editor-in-Chief:	Janice Walter
Contributing Editor:	Van Wolverton
Editors:	Mary Jacobson
	Linda Recktenwald
	Cecilia Crosby-Lampkin
	Tessa Gavron
Production Artists:	Karen Collins
	Julie Jefferson
	Kate Stites
Managing Editor:	Mark Kimbell
Circulation Manager:	Tammy Castleman
Editorial Director:	Jeff Yocom
Publishers:	Mark Crane
	Jon Pyles

Advertising: For information about advertising in Cobb Group journals, contact Tracee Bell Troutt at (800) 223-8720, extension 430.

Back Issues: To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $6.00 each, $8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you.

Bulk Sales: For information about bulk/group subscription sales, contact Larry Bolton at (800) 223-8720, extension 387.

Advisory Board: Earl Berry Jr.
Tina Covington
Marvin D. Livingood

choosing Save Setup As... from the File menu. Change the File Name entry to *DAILY.SET*, change the Description entry to *Daily backup--nonexecutables*, and press [Enter] to save the file.

Now you have two setup files you can use to back up files from the command line. Exit to the DOS prompt by pressing [Alt]F and then X.

Running MSBACKUP from the command line

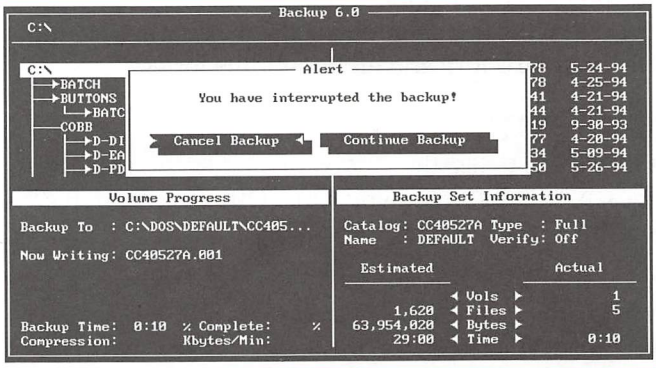
You make a monthly backup by entering the command

```
C:\>msbackup monthly /batch
```

at the DOS prompt. MSBACKUP will cycle through the opening dialog boxes and prompt you to insert the first diskette into the drive you specified in the setup file. When that diskette is full, MSBACKUP will prompt you to insert the second diskette into the drive. As each disk fills up, you'll be prompted to insert a disk. You don't need to press a key—just insert a disk.

If you want to interrupt the backup, just press [Esc] or [Enter]. MSBACKUP will show you the dialog box shown in Figure C. Press C to cancel the backup, or press B to continue the backup. When MSBACKUP backs up all the specified files, it will return you to the DOS prompt.

Figure C



You'll see the Alert box when you pause the backup by pressing [Esc] or [Enter].

Conclusion

If you use MSBACKUP to back up your files, you can make it even easier to use by employing the shortcut we showed you in this article. You simply use a setup file and the /BATCH switch to run the utility from the command line. ◆

UTILITY

VERSION 5.0 & 6.x

Deleting files from a directory and its subdirectories with MERGE.COM

David Reid, editor-in-chief of The Cobb Group's journals Inside AutoCAD *and* Exploring Windows NT, *co-authored this article.*

Here's a familiar scene: You're running out of disk space. You know your hard drive is littered with TMP and BAK files created by who-knows-what application. You think you can free up a fair amount of space by deleting these useless files. How do you do it?

First, you groan. Then you pour another cup of coffee and settle in for a long afternoon. You print the results of a couple of DIR commands that show you the location of all your TMP and BAK files in all the directories and subdirectories on the current drive:

```
dir /s *.tmp > prn
dir /s *.bak > prn
```

Then you painstakingly go through the printout, deleting files and checking off each directory as you issue commands like

```
del *.tmp
del *.bak
del 1stdir\*.tmp
del 1stdir\*.bak
del 2nddir\*.tmp
del 2nddir\*.bak
```

Now, wouldn't it be nice if the DEL command had an /S switch that let you delete the BAK and TMP files from the current directory and all its subdirectories. Commands like DIR, ATTRIB, and XCOPY provide this capability, so why not DEL? Unfortunately, we don't have an answer to that question, but we do have a solution to the deleting dilemma. Our solution comes in the form of a DEBUG utility that lets you feed a list of files generated by a DIR command into another DOS command that acts on each file in the list. You end up with a batch file that contains a list of commands similar to the list of DEL commands above.

In this article, we'll look at how you create a file list from the output of a DIR command. Then we'll present a DEBUG program, MERGE.COM, that converts the DIR command's output into a series of ready-to-run DOS commands. Finally, we'll create a batch file that will let you automate the process of deleting files in a directory and all its subdirectories.

Using DIR to create a file list

Our goal is to use the DIR command to create a list of files. Normally, when you run a DIR command, the output looks like this

```
Volume in drive C is WALT_6_2
Volume Serial Number is 16D8-2431
Directory of C:\GU

.           <DIR>          01-31-93 6:21a
..          <DIR>          01-31-93 6:21a
SHOW    EXE        34,317   06-19-91 9:52p
SNAP    EXE        42,416   10-29-91 4:35p
SAVE    EXE        27,060   10-29-91 4:37p
SAVE    BAK        27,060   10-29-91 4:37p
TEMP        <DIR>          01-31-93 7:45a
     6 file(s)         103,793 bytes
                   133,480,448 bytes free
```

As you can see, it contains extraneous information: The header shows disk and drive information and the footer shows file number and size information. Also, the DIR command lists the size and last modification date and time of each file. To turn this output into a file list, you must somehow remove this extra information.

Furthermore, the DIR command displays filenames using white space instead of a period between the file's base name and extension. This format is unacceptable as input to all other DOS commands.

Luckily, starting with DOS 5, Microsoft added the /B (Brief) switch to the DIR command. This switch tells DIR to omit the extraneous information and to output filenames only. In addition, this switch makes DIR output filenames in the standard *filename.ext* format. For example, when we type the command

```
C:\>dir c:\gu /b
```

the output from the previous example displays like this

```
SHOW.EXE
SNAP.EXE
SAVE.EXE
SAVE.BAK
TEMP
```

As you see, this output doesn't contain the usual header and footer information or any file size, date, or time information.

By default, the /B switch lists the names of both subdirectories and normal files. Usually, you'll want to eliminate the subdirectories from your file list. You can do so by adding the /A-D switch, which eliminates items whose attribute contains a directory bit.

You can also add the /S switch to tell DIR to extend the listing to include all files in directory trees branching from the starting directory. When you do so, DIR prefixes each entry in the resulting file list with its complete path. When we add the /A-D and /S switches to the command in our previous example

```
C:\>dir c:\gu /b /a-d /s
```

this is the result:

```
C:\GU\SHOW.EXE
C:\GU\SNAP.EXE
C:\GU\SAVE.EXE
C:\GU\SAVE.BAK
C:\GU\TEMP\VLIST-A.TMP
C:\GU\TEMP\VLIST-B.TMP
```

You can even use variations of the /O switch to arrange the directory listing in order by name, size, extension, or date. Once you've determined the form of the DIR command that will create the list you want, you can use DOS's > redirection operator to capture the list in a file. For example, we can redirect the file list above to FILE.LST by using the command

```
C:\>dir c:\gu /b /a-d /s > file.lst
```

At this point, we know how to create a file that contains the file list. Now all we need is a way to insert a command, such as DEL, before the name of each file. That's where the MERGE utility comes in. Let's create MERGE.COM now and discuss it in a moment.

Creating MERGE.COM

To create MERGE.COM, you create the script file MERGE.SCR. After creating the script file, you run the DEBUG utility, using the script file as input. When DEBUG runs the commands in MERGE.SCR, it creates MERGE.COM and then returns to the DOS prompt.

You can create the script file in the DOS Editor or any word processor that creates a text file without adding special formatting codes. Type the code exactly as it appears in Figure A. (Note that all the Os in the left column are the uppercase letter O; all the 0s in the right column are zeroes.) Be sure to leave a blank line before the RCX command—the script won't compile correctly if you omit it. After you've checked all the lines in MERGE.SCR for accuracy, save the file, exit to DOS, and enter the command

```
C:\>debug < merge.scr
```

to create the executable file MERGE.COM.

Figure A

```
N MERGE.COM
A       100
MOV     CX,1
MOV     DI,168
MOV     AH,3F
MOV     BX,0
MOV     DX,DI
INT     21
JB      14A
CMP     AX,CX
JNZ     14A
CMP     BYTE PTR [DI],A
JZ      106
CMP     BYTE PTR [DI],D
JZ      122
INC     DI
JMP     106
MOV     SI,81
CMP     BYTE PTR [SI],D
JNZ     137
MOV     DX,SI
CALL    14F
MOV     DX,117
CALL    14F
JMP     103
CMP     BYTE PTR [SI],3F
JNZ     142
CALL    157
INC     SI
JMP     125
MOV     DX,SI
CALL    14F
INC     SI
JMP     125
MOV     AX,4C00
INT     21
MOV     AH,40
MOV     BX,1
INT     21
RET
MOV     DI,168
CMP     BYTE PTR [DI],D
JZ      167
MOV     DX,DI
CALL    14F
INC     DI
JMP     15A
RET

RCX
68
W
Q
```

This DEBUG script creates MERGE.COM.

Using the MERGE utility

The MERGE utility merges the command on its command line with one line at a time from a text file you send it as input. We'll use the output of a DIR command as the text file, although you can use an existing text file or the output of any DOS command that creates a text file. You redirect the resulting output into a batch file that performs the command sequentially on an entire list of files. You use the MERGE utility in a command that has the form

```
text_cmd filespec | merge
    command ? > file.bat
```

where *text_cmd* is the DOS command that creates a text file or is the TYPE command, which retrieves a text file to the screen; *filespec* is the group of files you want to add to the file list; *command* is the command you want to execute with the files in the file list; and *file.bat* is the name of the batch file you'll use for storing the list of commands the MERGE utility creates.

At each point in its command line where a question mark (?) appears, the MERGE utility replaces the question mark with the current item in the file list. Continuing with our previous example, you can create a batch file that will delete the TMP and BAK files in the \GU directory by using the commands

```
C:\>dir gu\*.tmp /b /s /a-d | merge del ? > killfile.bat
```

and

```
C:\>dir gu\*.bak /b /s /a-d | merge del ? >> killfile.bat
```

When you run the two commands, you'll create KILLFILE.BAT, which contains three commands:

```
del C:\GU\TEMP\VLIST-B.TMP
del C:\GU\TEMP\VLIST-B.TMP
del C:\GU\SAVE.BAK
```

Now all you have to do to delete the files is run the KILLFILE batch file. Let's go one step further by creating a simple batch file, BDELETE.BAT, you can use to delete files from a directory and its subdirectories.

BDELETE.BAT

We show the batch file, BDELETE.BAT, in Figure B. You can create the file in the DOS Editor or any other word processing program that creates an ASCII text file without adding any formatting commands. Simply type the text as it appears in Figure B, save the file with the name BDELETE.BAT, and exit the program.

Figure B

```
@echo off
rem BDELETE.BAT deletes files you specify from the
rem current directory and all its subdirectories.
dir %1 /b /s /a-d | merge del ? > killfile.bat
call killfile.bat
del killfile.bat
```

You create BDELETE.BAT in the Editor or another ASCII text processor.

We made BDELETE.BAT as simple as possible. You might want to add commands that verify that you entered a valid file specification. Or you might add commands that type KILLFILE.BAT to the screen and ask you to confirm that you want to delete all the listed files.

The batch file starts with the @ECHO OFF command, which prevents the commands from displaying as the batch file runs, and two REM statements that show the name and purpose of the batch file.

The heart of the batch file rests in the next three commands. The first part of the DIR command retrieves a directory listing of the files that match the file specification you enter at the command line. The parameter %1 represents this file specification. We use the directory switch /B to strip out extraneous information, the /S switch to extend the directory listing to all the subdirectories of the specified directory, and the /A-D switch to remove the names of the subdirectories from the file list.

The segment of the DIR command following the pipe operator (¦) uses the MERGE.COM program to merge the DEL command with each filename in the file list. Then it uses the > operator to direct each newly created DEL *filespec* command into a batch file named KILLFILE.BAT.

The CALL command sends control to the KILLFILE.BAT batch file, which executes each DEL *filespec* command. Once KILLFILE.BAT executes the final DEL *filespec* command, control returns to the line in BDELETE.BAT following the point at which it branched out to the other batch file. The final command deletes our temporary batch file KILLFILE.BAT.

To use BDELETE.BAT, you simply enter a command in the form

```
bdelete filespec
```

For example, you can delete the TMP and BAK files from the \GU directory and its subdirectories by entering two commands:

```
C:\>bdelete gu\*.tmp
```

and

```
C:\>bdelete gu\*.bak
```

Of course, this batch file is much more useful when you want to delete every file on your system that has a particular file extension. For example, to delete all the TMP files in every directory on your C: drive, you enter the command

```
C:\>bdelete *.tmp
```

However, be very careful when you use this batch file—you don't want to inadvertently wipe out the wrong files.

Notes

In this article, we showed you how to use the MERGE utility with the DIR command to delete files. In "Copying Large Directories to More Than One Disk" on page 8, we use the utility to copy files. You can use MERGE with any DOS command that creates a text file, such as MEM, or with any text file whose information appears on individual lines. If you try creating a batch file that uses MERGE, simply place a REM command before the commands that CALL and DELETE the temporary batch file. Once you run the initial DIR/MERGE command, open the temporary batch file in the Editor or type it to the screen to make sure it contains the commands you want to execute.

Conclusion

DOS doesn't offer a built-in means of deleting all files having a common file specification from a directory and all its subdirectories. However, you can use the MERGE utility and batch file we presented in this article to perform this task. ◆

QUICK TIP

VERSION 5.0 & 6.x

Printing the output of DOS commands

In the accompanying article, we send the output of the DIR command to a file. You also can save the output of a number of other DOS commands by using the syntax

```
command > filename
```

Moreover, you can send the output of many DOS commands straight to the printer using a command in the form

```
command > printer
```

where *printer* is the generic or specific name of the printer (PRN, LPT1, or LPT2, for example). If you use a laser printer, you may have to send it a form feed in order to print the page if the command creates less than a full page of information.

In addition to the DIR command, you might want to save or print the output of other commands that produce text and listings. These commands include MEM, ATTRIB, CHKDSK, FC, HELP *command* (in DOS 5), FASTHELP *command* (in DOS 6.x), TREE, DOSKEY /M, and UNDELETE /LIST.

For example, you can print a list of all the deleted files on your system by using the command

```
C:\>undelete /list > prn
```

Then you can read through the list and undelete individual files by entering the UNDELETE command followed by the filename as it appears on the list, with a question mark as the first letter. To undelete CASTLE.BMP, you enter the command

```
C:\>undelete ?astle.bmp
```

When the Undelete utility prompts you to enter the first letter of the filename, type C and the file will be recovered. ◆

BATCH FILE TIP

VERSION 5.0 & 6.x

Preventing your batch files from displaying messages

When we want to prevent batch files from displaying DOS messages, we usually use the > redirection operator and the NUL device. For example, we use the NUL directive in the command

```
copy fileone filetwo > nul
```

to suppress the DOS message *1 file(s) copied*. (We use *fileone* and *filetwo* to represent any two files.) However, if the file named FILEONE doesn't exist, the command above will suppress the message *0 file(s) copied*, but it won't suppress the error message *File not found - FILEONE*. The reason: DOS commands send some messages to the *standard output device* and send error messages to the *standard error device*. The > redirection symbol affects only messages that go to the standard output device.

Sometimes you want to prevent your batch file from displaying all messages—even the error messages. Fortunately DOS introduced in Version 2.0 a useful, albeit little known, command you can use to suppress all screen messages: CTTY. We make extensive use of this command in the MULTCOPY.BAT batch file in "Copying Large Directories to More Than One Disk" on page 8. In this article, we'll introduce you to CTTY and show you how to use it in a batch file.

Introducing CTTY

The CTTY command lets you specify what device DOS will use as its console device—the device you use to input DOS commands and to which DOS sends all its output. The default console device is CON—the keyboard and monitor. You can specify an alternative device, such as AUX, COM1, or LPT1, if you want to run DOS from a device other than the console.

You can also use the NUL device with the CTTY command by entering the command

```
ctty nul
```

Doing so tells DOS to accept user input from and send all screen output to, in effect, nowhere. If you type the CTTY NUL command at the DOS command line, you won't be able to enter any subsequent commands, and you'll have to reboot your computer to regain control. Therefore, you should use the CTTY NUL command only within a batch file because a batch file automatically inputs a stream of commands.

Using CTTY in a batch file

You use a pair of CTTY commands to turn the screen display on and off. You turn the screen display off using the command CTTY NUL. You turn the screen display back on using the command CTTY CON. Always place a CTTY CON command at the end of a batch file in which you've used CTTY NUL to ensure that control returns to the console after the batch file ends.

Usually, you use these two commands in tandem—you place CTTY NUL before a command that creates a message you want to suppress, then you place CTTY CON immediately after that command. However, if you have several commands whose message you want to suppress, you can place a block of commands between the two CTTY commands.

You should be especially careful placing the CTTY NUL command in batch files that use GOTO commands to branch or loop to other sections of the batch file. Let's say that you place a single CTTY NUL command before a series of IF...GOTO statements, like this

```
ctty nul
if errorlevel 5 goto :ERROR
if errorlevel 4 goto :CHANGE
if errorlevel 0 goto :START_COPY
```

and that the :ERROR and :CHANGE sections use ECHO statements to display messages. In this case, you'll need to turn on the screen display by placing a CTTY CON command on the line following the :ERROR and :CHANGE labels. However, you might want to suppress the messages in the :START_COPY section. In this case, the CTTY NUL statement you used before the IF statements will continue to do the job.

You might want to use a CALL command to pass control to a second batch file while your batch file is running. If you do so, any CTTY NUL command that's in effect when the second batch file runs will suppress any messages the second batch file's commands create. If you want to display the second batch file's messages, you place a CTTY CON command immediately before the CALL command.

After you've added a few CTTY NUL and CTTY CON statements to a batch file, you might lose track of which device—NUL or CON—is in control at any given time. If this happens, don't panic. You can debug your batch file by adding CTTY CON commands to your batch file wherever you think you might need one, followed by ECHO statements that tell you what part of the batch file is currently processing. Then, place the REM command in front of individual CTTY CON statements while you're testing your batch file. Eventually, you'll determine exactly where you need to place CTTY statements. ◆

VERSION 5.0 & 6.x

Copying large directories to more than one disk

How many times have you wanted to copy all the files from a large directory and its subdirectories onto a floppy disk? Undoubtedly, you first tried a COPY command like this one:

```
C:\>copy dos\*.* b:
```

And just as certainly, you discovered that DOS didn't copy all the files onto a single disk. Moreover, there isn't an easy way to determine which files DOS copied or to copy the remaining files.

In this article, we're going to show you how to solve this dilemma. First, we'll show you how to copy large directories using a couple of simple DOS commands at the DOS prompt. Then we'll show you a batch file that not only copies the files but prompts you for new disks and tells you when all the files are copied. In addition, the batch file erases all the files from each floppy disk before it begins copying files. The batch file uses the MERGE utility we created in the article "Deleting Files from a Directory and its Subdirectories with MERGE.COM" on page 3.

Copying large directories from the command line

DOS provides two commands that copy files: COPY and XCOPY. Fortunately, XCOPY provides the capability we're seeking. You can use the XCOPY command to copy all the files from a large directory to more than one disk. First, you use the ATTRIB command to set the archive bit of all the files in the directory and its subdirectories like this:

```
attrib filespec +a /s
```

where *filespec* is the name of a group of files you specify with wildcards, such as *.TIF. You can include in *filespec* a drive or directory specification (or both), such as DOS*.*. The +A directive sets the archive attribute. The /S switch extends processing to files in all the directories in the specified path.

After you set the files' archive bits, you use the XCOPY command with the /M and /S switches to copy the file. The /M switch tells XCOPY to copy files that have the archive bit set and then turn off a file's archive bit as it copies the file. The /S switch tells XCOPY to copy directories and subdirectories, except the empty ones. You enter the XCOPY command like this

```
xcopy source target /m /s
```

where *source* contains the drive, directory, and/or group of files to copy and *target* is the drive, directory, and/or filename to which you're copying the files.

Let's use this technique to copy all the files from the DOS directory and its subdirectories to a disk in the B drive. First, set the archive bits of all the files:

```
C:\>attrib dos\*.* +a /s
```

Next, issue the command to copy the files:

```
C:\>xcopy dos b: /m /s
```

DOS will begin copying the files in the order they're stored on your hard disk. When the first disk in drive B is full, DOS presents the message *Insufficient disk space*. At this point you can put a second disk in the B drive and press [F3] to retype the XCOPY command on the command line. Press [Enter] and DOS will resume copying. When DOS finishes copying all the files, it presents the *# file(s) copied* message.

A batch file for copying large directories

Copying lots of files from the command line is easy enough, so you might wonder why you'd go to the trouble of creating a batch file that essentially does the same thing. The batch file provides some amenities that the simple command-line method doesn't, including:

- relieving you of the need to remember the ATTRIB and XCOPY commands' syntax
- deleting all the files from the floppy disk before beginning to copy files
- copying the largest files first so the floppy disks are packed as fully as possible
- prompting you to exchange disks when the current one is full

Unfortunately, there is one thing that neither the batch file nor the command-line method can do: copy a file that's larger than the capacity of your floppy disk. In order to copy such a large file, you can use a third-party compression program such as PKZIP or, if you have DOS 6.x, you can use DoubleSpace to compress a floppy disk.

Now let's create the batch file—we'll discuss how it works in a moment. Using the DOS Editor or another word processing program that creates text files without adding any formatting codes, enter the commands shown in Figure A. If you use DOS 5, you enter the commands shown in Figure B for the :DELETE_B section.

Figure A

```
@echo off
rem MULTCOPY.BAT lets you copy a large group of files
rem or files from a large directory onto several disks
rem and prompts you when the disk is full.

:CHECK_IF_FILE_EXISTS
if exist %1 goto :START
if exist %1\nul goto :START
goto :ERROR

:START
rem Sets the archive bits of the specified files.
if exist %1 attrib %1 +a /s
if exist %1\nul attrib %1\*.* +a /s
echo Place a disk in drive B: and
pause

:DELETE_B
rem This DOS 6.x version uses DELTREE to delete everything
rem from B: disk. See Figure B for DOS 5 commands to use
rem in this section.
echo Deleting files from drive B:
attrib b: /s -r -h -s
ctty nul
deltree /y b:
ctty con

:COPYING
rem Uses MERGE.COM to create MULTICOP.BAT, which copies files
dir %1 /aa /o-s/ s /b ¦ merge xcopy /m ? b: > multicop.bat
echo Copying files!
ctty nul
call multicop.bat

:IS_IT_DONE?
rem Checks whether there are more files to copy
dir %1 /aa /b > multicop.01
copy multicop.01 multicop.02
if not exist multicop.02 goto :FINISH
ctty con
echo This disk is full. Please label it
echo and insert another disk. Then
pause
goto :DELETE_B

:FINISH
ctty con
echo All your files are copied!
goto :END

:ERROR
echo You entered the name of a file or directory
echo that does not exist. Please try again.

:END
del multicop.*
ctty con
```

MULTCOPY.BAT copies a large group of files or files from a large directory to multiple floppy disks.

Figure B

```
:DELETE_B
rem Uses MERGE.COM to delete everything
rem from B: disk in DOS 5.
echo Deleting files from drive B:
attrib b: /s -r -h -s
ctty nul
echo y ¦ del b:*.*
dir b: /s /a-d /b ¦ merge del ? > delete-b.bat
dir b: /s /ad /b ¦ merge rd ? >> delete-b.bat
copy delete-b.bat delete-b.txt
if not exist delete-b.txt goto :COPYING
call delete-b.bat
del delete-b.*
ctty con
```

In DOS 5, use these commands for the :DELETE_B section.

How MULTCOPY.BAT works

MULTCOPY.BAT begins with the @ECHO OFF command, which prevents the batch file's commands from displaying as the batch file runs. The REM commands hold the batch file's name and purpose. Let's look at each section of the batch file.

CHECK_IF_FILE_EXISTS

The commands in :CHECK_IF_FILE_EXISTS test whether you enter a valid filename or directory name when you run the batch file. The first command tests for filenames; the second command uses the NUL directive to test for the name of a directory. If you enter neither a valid filename nor a valid directory name, the batch file branches to the :ERROR section. (You can use a similar block of statements in any batch file to trap for a valid file or directory name entered with the batch file name.)

:START

The next section, :START, sets the archive bits of the files. The first IF statement sets the archive bit for filenames; the second IF statement sets the archive bit for the files in a directory and its subdirectories. Next, this section prompts you to place a floppy disk in the B drive. (You can specify another drive letter here if

you'd rather copy files elsewhere. Simply specify another letter at those places marked in red in the batch file.) Then the PAUSE command stops the batch file until you press any key to continue.

:DELETE_B

The :DELETE_B section is responsible for deleting all the files from the disk in drive B. :DELETE_B starts by displaying the message *Deleting files from drive B:* and then removes the attributes from all the files on the disk in drive B. The CTTY NUL command turns off the screen display so you don't see the messages created by the commands that delete the files. (We discuss how to use this command in the article "Preventing Your Batch Files from Displaying Messages" on page 7.) The version of :DELETE_B in Figure A uses the DELTREE command to delete all the files from the disk. We use the /Y switch to suppress the prompts that ask you to verify that you want to delete each file.

The DOS 5 :DELETE_B section, shown in Figure B, is a little more complicated than the DOS 6.x version. DOS 5 doesn't offer the DELTREE command, so we build one using the MERGE.COM utility we created in "Deleting Files from a Directory and its Subdirectories with MERGE.COM." After displaying the message, removing the attributes from all the files on the disk, and suppressing the display of screen messages, the DOS 5 version of :DELETE_B uses the command

```
echo y ¦ del b:*.*
```

to delete all the files from the root directory of the floppy disk. The next two commands use DIR commands to check the floppy disk for the presence of subdirectories and use the MERGE utility to create a batch file that will delete all the files from those subdirectories and then remove the subdirectories. Next,

```
copy delete-b.bat delete-b.txt
```

tests whether the two previous commands found subdirectories. If the commands didn't find subdirectories, the DELETE-B.BAT file will contain zero bytes. Since DOS won't let you copy a zero-byte file to another file, you can test whether there are subdirectories by seeing whether the DELETE-B.TXT file exists. If it doesn't, it means the floppy disk is now empty and ready for copying files. If DELETE-B.TXT exists, it means that the batch file must remove the subdirectories, which it then proceeds to do by calling the batch file DELETE-B.BAT. Once DELETE-B.BAT deletes the files from the subdirectories and removes the subdirectories, control returns to MULTCOPY.BAT, which then deletes the two DELETE-B files.

:COPYING

Next comes the heart of the batch file, the :COPYING section. Let's take a close look at the main command in this section. The first segment

```
dir %1 /aa /o-s/ s /b
```

retrieves a directory listing of the file specification or directory name you enter when you run the batch file. It retrieves the names of files whose archive bit is set (/AA) in order by size from largest to smallest (/O-S). It checks the current directory and its subdirectories (/S) and presents the filenames in brief format (/B), which includes only the drive letter, directory names, and filenames.

The second segment of the main command

```
¦ merge xcopy /m ? b: > multicop.bat
```

merges each line of the directory listing with the rest of the XCOPY command, replacing the question mark with each filename in the directory listing. It creates a list of commands in the form

```
xcopy /m drive-dir-filename b:
```

The > redirection operator places the XCOPY commands, one per line, in the MULTICOP batch file.

Next, the batch file shows a message and then turns off the screen display. (The first time you run the batch file, you might want to place the REM command at the beginning of the CTTY NUL command to see how the batch file operates.) The command

```
call multicop.bat
```

starts the internal batch file, which executes its series of XCOPY commands. As XCOPY copies a file to the B: disk, it uses the /M switch to remove the file's archive attribute. When MULTICOP.BAT has placed all the files it can fit on the B: disk, control returns to MULTCOPY.BAT at the :IS_IT_DONE? section.

:IS_IT:DONE?

In this section, the command

```
dir %1 /aa /b > multicop.01
```

retrieves a listing of only the files whose archive attribute is still set—this directory listing doesn't include the files that are already copied. The output of this DIR command is redirected into a text file, MULTICOP.01. If MULTICOP.01 is empty, it means there aren't any more files to copy. If this is the case, the next command can't copy the file's con-

tents to MULTICOP.02, which won't exist. Thus, the command

```
if not exist multicop.02 goto :FINISH
```

sends the batch file to the :FINISH section, which alerts you that all the files are copied.

However, if MULTICOP.02 does exist, you see the message that begins *This disk is full*. The batch file pauses so you can insert a new disk. When you press any key to continue, the batch file loops back to the :DELETE_B section and the process begins again from the point at which it deletes the files from the disk in drive B. Eventually, the batch file will run out of files to copy and will branch to the :FINISH section, which branches to the :END section.

:END

The last section deletes the temporary files the batch file created. We use the CTTY CON command at the end as insurance, in case we left a CTTY NUL statement without a corresponding statement to return to the default input/output device when the batch file ends.

Using the batch file

To use the MULTCOPY batch file, you simply enter the name of the batch file followed by the name of the directory or the file specification of the group of files you want to copy. For example, to copy the DOS directory, you use the command

```
C:\>multcopy dos
```

If you want to copy all the *.TIF files on your system to a disk in drive B, you use the command

```
C:\>multcopy *.tif
```

You can also specify a group of files in a specific directory. For example, use the following command to copy all the files in the WINDOWS directory and subdirectories that have a DLL extension:

```
C:\>multcopy windows\*.dll
```

Once you start the batch file, you'll see messages that let you know how the batch file is proceeding. With the first floppy disk, the batch file might take a minute or two to copy the files. When you see the *Copying files!* message, just hang in there—the batch file isn't stuck, it just has a lot of files to copy and try to fit on the disk.

An example

Let's take our batch file on a quick run. In this example, we'll look at how MULTCOPY.BAT copies the contents of the GU directory and its subdirectories. Place a disk in drive B and start the batch file by entering the command

```
C:\>multcopy gu
```

Press any key when you see the prompt

```
Place a disk in drive B and
Press any key to continue . . .
```

Next, you see the message

```
Deleting files from drive B:
```

as the batch file removes all files and directories from the disk.

Next, the :COPYING section runs a DIR /S /A-D /B command through the MERGE filter to create the MULTICOP batch file containing these commands

```
xcopy /m C:\GU\SHOW.EXE b:
xcopy /m C:\GU\SNAP.EXE b:
xcopy /m C:\GU\SAVE.EXE b:
xcopy /m C:\GU\SAVE.BAK b:
xcopy /m C:\GU\TEMP\VLIST-A.TMP b:
xcopy /m C:\GU\TEMP\VLIST-B.TMP b:
```

Then you see the message

```
Copying files!
```

and the MULTICOP batch file runs, executing each XCOPY command in turn. The first five files completely fill the floppy disk, so when the :IS_IT_DONE? section checks the archive bits of the files in the GU directory, it discovers that one file's bit is still set. Therefore, you see the prompt

```
This disk is full. Please label it
and insert another disk. Then
Press any key to continue . . .
```

Once you replace the floppy disk and press a key, you again see the *Deleting* message, followed by the *Copying files!* message. Finally, the batch file determines that there are no more files to copy and you see the *All your files are copied!* message. At this point, the batch file ends.

Conclusion

You can use the XCOPY command to copy large directories or large groups of files to a floppy disk. In this article, we showed you how to use XCOPY to do just that from the command line and from a batch file. ◆

SECOND CLASS MAIL

Microsoft Technical Support
(206) 454-2030

LETTERS

EDIT.COM can't find QBASIC.EXE

I was trying to free up some space on my hard disk, so I moved QBasic to a floppy disk, since I never use it. Later, I tried to open the DOS Editor and received the error message

```
Can not find file QBASIC.EXE
```

My computer wouldn't let me use the Editor, so I put all the QBasic files back into my DOS directory. Now the Editor works, but I'm perplexed. What does QBasic have to do with the DOS Editor?

Walt Wicker
Rock Hills, Kentucky

QBasic has a lot to do with the DOS Editor. As you discovered, you can't run the DOS Editor unless QBasic is on your system. QBasic must be present because it, not EDIT.COM, holds the code that runs the DOS Editor. When you execute the 413-byte EDIT.COM command, it simply invokes the 190 Kb QBASIC.EXE command (249 Kb in DOS 5) with the /EDITOR switch. In fact, you could open the DOS Editor by entering the command

```
C:\>qbasic /editor
```

In order to open the DOS Editor with the EDIT.COM command, QBasic must be in the current directory, in a directory listed in the PATH statement, or in the same directory as EDIT.COM. Incidentally, the DOS 6 Help system also uses the QBasic interpreter. Therefore, if you want to use the Editor and, in DOS 6, the Help system, you shouldn't try to save hard disk space by removing QBASIC.EXE from your system.

Why do I get a *Sharing violation* message with MEMORY.BAT?

I ran into some problems with MEMORY.BAT, which appeared in your May 1994 article "Tracking the Results of Changes to Your System Configuration." When I ran the batch file, the following message appeared:

```
C:\Z.ZZZ is currently being printed
```

There was nothing but silence. After a moment, the next message appeared:

```
Sharing violation reading drive C
Abort, Retry, Fail?
```

Fearing that I had created a command error, I typed the letter *A* for Abort. Well, the printer came on like gangbusters and printed the report. However, the batch file didn't delete Z.ZZZ after it printed. What happened?

Porter L. Baymon
Oakland, California

First, there's nothing wrong with your batch file or your system. However, the *Sharing violation* message indicates you have SHARE loaded or you're on a network. SHARE puts a lock on an open file, meaning that no other user or command can access the file until DOS recognizes that it's closed. In this case, SHARE locks Z.ZZZ after the file is sent to the printer. The last line in the batch file tries to delete Z.ZZZ but it can't because Z.ZZZ is locked at this point. Unfortunately, the file can't print until you respond to the error message.

As you discovered, when you pressed A, the document printed—that's because the command you aborted was the command to delete Z.ZZZ. Fortunately, having Z.ZZZ on your system isn't a problem because the next time you run the batch file, it will overwrite Z.ZZZ.

You can avoid having to press A at the *Abort, Retry, Fail?* message by adding the /F switch near the end of the SHELL statement in your CONFIG.SYS file. The /F switch automatically passes F (for Fail) to the *Abort, Retry, Fail?* message and returns you to the DOS prompt. (We discussed the /F switch in the October 1993 article "Bypassing the *Abort, Retry, Fail?* Error Message.") ◆

Printed in the USA
This journal is printed on recycled and recyclable paper.
50% recycled fiber including 10% post-consumer waste paper